## APPENDIX

```
This is the function sobol for generating Sobol points. It returns the
  n-th d-dimensional Sobol point. (The point (0,0, ..., 0) is skipped.)
  The point is implicitly returned through the array x. The function
  sobseq from Numerical Recipes, 1992 was used as a basis, but
  changed significantly to accommodate the parallel distributed approach.
  The function sobol can generate Sobol points skipping an initial
  part of the sequence. The constant MAXDIM, see Numerical Recipes,
  is extended to 360 which required adding more initializing data to
  the arrays ip (the primitive polynomials),
  mdeg (their degrees), and iv (the initial direction numbers).
  polynomial x is used to generate the first coordinate of Sobol points.
  *****************************
#include "nrutil.h"
#define MAXBIT 30
#define MAXDIM 360
extern int d;
         /* actual dimension of the points */
extern double *x;
            /* This returns implicitly the n-th Sobol point in x */
woid sobol(int n)
Ξ{
 int j,k,l;
 unsigned long i, im, ipp;
 static double fac;
static unsigned long in,ix[MAXDIM+1],*iu[MAXBIT+1];
static unsigned long mdeg[MAXDIM+1]={0,MAXBIT,1,2,3,3,4,4,
                5,5,5,5,5,5,
                6,6,6,6,6,6,
Ħ
                7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,
/*16*/
                    9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,
                9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,
                9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,
          /*20*/
       0200
       12,12,12
       };
```

void sobol(int n)

20

```
/* The first coordinate is set to MAXBIT, but it is never used */
 static unsigned long ip[MAXDIM+1]={0,MAXBIT,0,1,1,2,1,4,
/*deg 5*/
                          2,13,7,14,11,4,
/*deg 6*/
                          1,16,13,22,19,25,
/*deg 7*/
              1,32,4,8,7,56,14,28,19,50,21,42,31,62,37,41,55,59,
/*deg 8*/
           14,56,21,22,38,47,49,50,52,67,70,84,97,103,115,122,
/*deg 9*/
       8,13 ,16 ,22 ,25,44 ,47 ,52 ,55 ,59 ,62 ,67 ,74 ,81 ,82 ,87 ,91 ,94,
   103,104,109,122,124,137,138,143,145,152,157,167,173,176,181,182,185,
   191,194,199,218,220,227,229,230,234,236,241,244,253,
/*deg 10*/
          4,13,19,22,50,55,64,69,98,107,115,121,127,134,140,145,152,
     158, 161, 171, 181, 194, 199, 203, 208, 227, 242, 251, 253, 265, 266, 274, 283, 289, 295,
     301,316,319,324,346,352,361,367,382,395,398,400,412,419,422,426,428,433,
     446,454,457,472,493,505,508,
/*deg11*/ 2,11,21,22,35,49,50,56,61,70,74,79,84,88,103,104,112,115,117,122,134,
     137,146,148,157,158,162,164,168,173,185,186,191,193,199,213,214,220,227,
     236,242,251,256,259,265,266,276,292,304,310,316,319,322,328,334,339,341,
     345,346,362,367,372,375,376,381,385,388,392,409,415,416,421,428,431,434,
     439,446,451,453,457,458,471,475,478,484,493,494,499,502,517,518,524,527,
     555,560,565,569,578,580,587,589,590,601,607,611,614,617,618,625,628,635,
     641,647,654,659,662,672,675,682,684,689,695,696,713,719,724,733,734,740,
     747,749,752,755,762,770,782,784,787,789,793,796,803,805,810,815,824,829,
     830,832,841,847,849,861,871,878,889,892,901,908,920,923,942,949,950,954,
     961,968,971,973,979,982,986,998,1001,1010,1012,
//:*deg12*/  41,52,61,62,76,104,117,131,143,145,157,167,171,176,181,194,217,236,
    239,262,283,286,307};
 static unsigned long iv [MAXDIM*MAXBIT+1] = {
     /* 40 elements in a row */
/*end 1 */
/*end 2 */
1,5,7,7,5,1,3,3,7,5,5,7,7,1,3,3,7,5,1,1,5,3,3,1,7,5,1,3,3,7,5,1,1,5,7,7,5,1,3,3,
1,5,7,7,5,1,3,3,7,5,5,7,7,1,3,3,7,5,1,1,5,3,3,1,7,5,1,3,3,7,5,1,1,5,7,7,5,1,3,3,
1,5,7,7,5,1,3,3,7,5,5,7,7,1,3,3,7,5,1,1,5,3,3,1,7,5,1,3,3,7,5,1,1,5,7,7,5,1,3,3,
1,5,7,7,5,1,3,3,7,5,5,7,7,1,3,3,7,5,1,1,5,3,3,1,7,5,1,3,3,7,5,1,1,5,7,7,5,1,3,3,
1,5,7,7,5,1,3,3,7,5,5,7,7,1,3,3,7,5,1,1,5,3,3,1,7,5,1,3,3,7,5,1,1,5,7,7,5,1,3,3,
1,5,7,7,5,1,3,3,7,5,5,7,7,1,3,3,7,5,1,1,5,3,3,1,7,5,1,3,3,7,5,1,1,5,7,7,5,1,3,3,
1,5,7,7,5,1,3,3,7,5,5,7,7,1,3,3,7,5,1,1,5,3,3,1,7,5,1,3,3,7,5,1,1,5,7,7,5,1,3,3,
1,5,7,7,5,1,3,3,7,5,5,7,7,1,3,3,7,5,1,1,5,3,3,1,7,5,1,3,3,7,5,1,1,5,7,7,5,1,3,3,
1,5,7,7,5,1,3,3,7,5,5,7,7,1,3,3,7,5,1,1,5,3,3,1,7,5,1,3,3,7,5,1,1,5,7,7,5,1,3,3,
                                                         /*end 3
1,15,11,5,3,1,7,9,13,11,1,3,7,9,5,13,13,11,3,15,5,3,15,7,9,13,9,1,11,7,5,15,1,15,11,5,3,1,7,9
```

```
1,15,11,5,3,1,7,9,13,11,1,3,7,9,5,13,13,11,3,15,5,3,15,7,9,13,9,1,11,7,5,15,1,15,11,5,3,1,7,9,
1,15,11,5,3,1,7,9,13,11,1,3,7,9,5,13,13,11,3,15,5,3,15,7,9,13,9,1,11,7,5,15,1,15,11,5,3,1,7,9,
1,15,11,5,3,1,7,9,13,11,1,3,7,9,5,13,13,11,3,15,5,3,15,7,9,13,9,1,11,7,5,15,1,15,11,5,3,1,7,9,
1,15,11,5,3,1,7,9,13,11,1,3,7,9,5,13,13,11,3,15,5,3,15,7,9,13,9,1,11,7,5,15,1,15,11,5,3,1,7,9,
1,15,11,5,3,1,7,9,13,11,1,3,7,9,5,13,13,11,3,15,5,3,15,7,9,13,9,1,11,7,5,15,1,15,11,5,3,1,7,9,
1,15,11,5,3,1,7,9,13,11,1,3,7,9,5,13,13,11,3,15,5,3,15,7,9,13,9,1,11,7,5,15,1,15,11,5,3,1,7,9,
  /*end 4 */
1,17,13,7,15,9,31,9,3,27,15,29,21,23,19,11,25,7,13,17,1,25,29,3,31,11,5,23,27,19,21,5,1,17,13,
7,15,9,31,9,
7,15,9,31,9,
1,17,13,7,15,9,31,9,3,27,15,29,21,23,19,11,25,7,13,17,1,25,29,3,31,11,5,23,27,19,21,5,1,17,13,
7,15,9,31,9,
7,15,9,31,9,
7, 15, 9, 31, 9,
7,15,9,31,9,
📴, 17, 13, 7, 15, 9, 31, 9, 3, 27, 15, 29, 21, 23, 19, 11, 25, 7, 13, 17, 1, 25, 29, 3, 31, 11, 5, 23, 27, 19, 21, 5, 1, 17, 13,
遺, 15 , 9 , 31 , 9 ,
1, 17, 13, 7, 15, 9, 31, 9, 3, 27, 15, 29, 21, 23, 19, 11, 25, 7, 13, 17, 1, 25, 29, 3, 31, 11, 5, 23, 27, 19, 21, 5, 1, 17, 13,
7,15,9,31,9,
慧, 17, 13, 7, 15, 9, 31, 9, 3, 27, 15, 29, 21, 23, 19, 11, 25, 7, 13, 17, 1, 25, 29, 3, 31, 11, 5, 23, 27, 19, 21, 5, 1, 17, 13,
□,15,9,31,9,
             /*end 5 */
退,51,61,43,51,59,47,57,35,53,19,51,61,37,33,7,5,11,39,63,27,17,15,23,29,3,21,13,31,25,9,49,33,
19,29,11,19,27,15,25,
1,51,61,43,51,59,47,57,35,53,19,51,61,37,33,7,5,11,39,63,27,17,15,23,29,3,21,13,31,25,9,49,33,
<sup>2</sup>19,29,11,19,27,15,25,
🗓,51,61,43,51,59,47,57,35,53,19,51,61,37,33,7,5,11,39,63,27,17,15,23,29,3,21,13,31,25,9,49,33,
119,29,11,19,27,15,25,
4,51,61,43,51,59,47,57,35,53,19,51,61,37,33,7,5,11,39,63,27,17,15,23,29,3,21,13,31,25,9,49,33,
19,29,11,19,27,15,25,
量,51,61,43,51,59,47,57,35,53,19,51,61,37,33,7,5,11,39,63,27,17,15,23,29,3,21,13,31,25,9,49,33,
(19,29,11,19,27,15,25,
諡,51,61,43,51,59,47,57,35,53,19,51,61,37,33,7,5,11,39,63,27,17,15,23,29,3,21,13,31,25,9,49,33,
19,29,11,19,27,15,25,
1,51,61,43,51,59,47,57,35,53,19,51,61,37,33,7,5,11,39,63,27,17,15,23,29,3,21,13,31,25,9,49,33,
19,29,11,19,27,15,25,
1,51,61,43,51,59,47,57,35,53,19,51,61,37,33,7,5,11,39,63,27,17,15,23,29,3,21,13,31,25,9,49,33,
19,29,11,19,27,15,25,
1,51,61,43,51,59,47,57,35,53,19,51,61,37,33,7,5,11,39,63,27,17,15,23,29,3,21,13,31,25,9,49,33,
19,29,11,19,27,15,25, /*end 6 */
1,85,67,49,125,25,109,43,89,69,113,47,55,97,3,37,83,103,27,13,33,115,41,79,17,29,119,75,73,105
,7,59,65,21,3,113,61,89,45,107,
1,85,67,49,125,25,109,43,89,69,113,47,55,97,3,37,83,103,27,13,33,115,41,79,17,29,119,75,73,105
,7,59,65,21,3,113,61,89,45,107,
1,85,67,49,125,25,109,43,89,69,113,47,55,97,3,37,83,103,27,13,33,115,41,79,17,29,119,75,73,105
,7,59,65,21,3,113,61,89,45,107,
1,85,67,49,125,25,109,43,89,69,113,47,55,97,3,37,83,103,27,13,33,115,41,79,17,29,119,75,73,105
,7,59,65,21,3,113,61,89,45,107,
1,85,67,49,125,25,109,43,89,69,113,47,55,97,3,37,83,103,27,13,33,115,41,79,17,29,119,75,73,105
,7,59,65,21,3,113,61,89,45,107,
1,85,67,49,125,25,109,43,89,69,113,47,55,97,3,37,83,103,27,13,33,115,41,79,17,29,119,75,73,105
,7,59,65,21,3,113,61,89,45,107,
```

```
1,85,67,49,125,25,109,43,89,69,113,47,55,97,3,37,83,103,27,13,33,115,41,79,17,29,119,75,73,105
,7,59,65,21,3,113,61,89,45,107,
1,85,67,49,125,25,109,43,89,69,113,47,55,97,3,37,83,103,27,13,33,115,41,79,17,29,119,75,73,105
,7,59,65,21,3,113,61,89,45,107,
1,85,67,49,125,25,109,43,89,69,113,47,55,97,3,37,83,103,27,13,33,115,41,79,17,29,119,75,73,105
,7,59,65,21,3,113,61,89,45,107, /*end 7 */
49, 15, 213, 135, 253, 191, 155, 175, 63, 47, 7, 23, 39,
1,255,79,147,141,89,173,43,9,25,115,97,19,97,197,101,255,29,203,65,195,177,105,17,47,169,109,1
49, 15, 213, 135, 253, 191, 155, 175, 63, 47, 7, 23, 39,
1,255,79,147,141,89,173,43,9,25,115,97,19,97,197,101,255,29,203,65,195,177,105,17,47,169,109,1
49, 15, 213, 135, 253, 191, 155, 175, 63, 47, 7, 23, 39,
1,255,79,147,141,89,173,43,9,25,115,97,19,97,197,101,255,29,203,65,195,177,105,17,47,169,109,1
49, 15, 213, 135, 253, 191, 155, 175, 63, 47, 7, 23, 39,
1,255,79,147,141,89,173,43,9,25,115,97,19,97,197,101,255,29,203,65,195,177,105,17,47,169,109,1
49, 15, 213, 135, 253, 191, 155, 175, 63, 47, 7, 23, 39,
1,255,79,147,141,89,173,43,9,25,115,97,19,97,197,101,255,29,203,65,195,177,105,17,47,169,109,1
49, 15, 213, 135, 253, 191, 155, 175, 63, 47, 7, 23, 39,
49, 15, 213, 135, 253, 191, 155, 175, 63, 47, 7, 23, 39,
1,255,79,147,141,89,173,43,9,25,115,97,19,97,197,101,255,29,203,65,195,177,105,17,47,169,109,1
49,15,213,135,253,191,155,175,63,47,7,23,39,
運, 255, 79, 147, 141, 89, 173, 43, 9, 25, 115, 97, 19, 97, 197, 101, 255, 29, 203, 65, 195, 177, 105, 17, 47, 169, 109, 1
29, 15, 213, 135, 253, 191, 155, 175, 63, 47, 7, 23, 39,
                                               /*end 8 */
 嵭,393,167,333,473,469,101,21,451,229,247,297,403,497,61,361,
፝፟ቜ፞, 257, 465, 439, 177, 321, 181, 225, 235, 103, 411, 233, 59, 353, 329, 463, 385, 111, 475, 451, 263, 19, 249, 275, 36
£9,393,167,333,473,469,101,21,451,229,247,297,403,497,61,361,
让, 257, 465, 439, 177, 321, 181, 225, 235, 103, 411, 233, 59, 353, 329, 463, 385, 111, 475, 451, 263, 19, 249, 275, 36
9,393,167,333,473,469,101,21,451,229,247,297,403,497,61,361,
<u>1</u>,257,465,439,177,321,181,225,235,103,411,233,59,353,329,463,385,111,475,451,263,19,249,275,36
3,393,167,333,473,469,101,21,451,229,247,297,403,497,61,361,
🗓, 257, 465, 439, 177, 321, 181, 225, 235, 103, 411, 233, 59, 353, 329, 463, 385, 111, 475, 451, 263, 19, 249, 275, 36
寶, 393, 167, 333, 473, 469, 101, 21, 451, 229, 247, 297, 403, 497, 61, 361,
<u>4</u>, 257, 465, 439, 177, 321, 181, 225, 235, 103, 411, 233, 59, 353, 329, 463, 385, 111, 475, 451, 263, 19, 249, 275, 36
 ,393,167,333,473,469,101,21,451,229,247,297,403,497,61,361,
運, 257, 465, 439, 177, 321, 181, 225, 235, 103, 411, 233, 59, 353, 329, 463, 385, 111, 475, 451, 263, 19, 249, 275, 36
👰, 393, 167, 333, 473, 469, 101, 21, 451, 229, 247, 297, 403, 497, 61, 361,
1,257,465,439,177,321,181,225,235,103,411,233,59,353,329,463,385,111,475,451,263,19,249,275,36
9,393,167,333,473,469,101,21,451,229,247,297,403,497,61,361,
1,257,465,439,177,321,181,225,235,103,411,233,59,353,329,463,385,111,475,451,263,19,249,275,36
9,393,167,333,473,469,101,21,451,229,247,297,403,497,61,361,
/*end 9 */
1,771,721,1013,759,835,949,113,929,615,157,39,761,169,983,657,647,581,505,833,139,147,203,81,3
37,829,989,375,365,131,215,733,451,447,177,57,471,979,197,251,
1,771,721,1013,759,835,949,113,929,615,157,39,761,169,983,657,647,581,505,833,139,147,203,81,3
37,829,989,375,365,131,215,733,451,447,177,57,471,979,197,251,
1,771,721,1013,759,835,949,113,929,615,157,39,761,169,983,657,647,581,505,833,139,147,203,81,3
37,829,989,375,365,131,215,733,451,447,177,57,471,979,197,251,
1,771,721,1013,759,835,949,113,929,615,157,39,761,169,983,657,647,581,505,833,139,147,203,81,3
37,829,989,375,365,131,215,733,451,447,177,57,471,979,197,251,
1,771,721,1013,759,835,949,113,929,615,157,39,761,169,983,657,647,581,505,833,139,147,203,81,3
37,829,989,375,365,131,215,733,451,447,177,57,471,979,197,251,
1,771,721,1013,759,835,949,113,929,615,157,39,761,169,983,657,647,581,505,833,139,147,203,81,3
37,829,989,375,365,131,215,733,451,447,177,57,471,979,197,251,
1,771,721,1013,759,835,949,113,929,615,157,39,761,169,983,657,647,581,505,833,139,147,203,81,3
37,829,989,375,365,131,215,733,451,447,177,57,471,979,197,251,
```



```
37,829,989,375,365,131,215,733,451,447,177,57,471,979,197,251,
1,771,721,1013,759,835,949,113,929,615,157,39,761,169,983,657,647,581,505,833,139,147,203,81,3
37,829,989,375,365,131,215,733,451,447,177,57,471,979,197,251,
/*end 10 */
1,1285,823,727,267,833,471,1601,1341,913,1725,2021,1905,375,893,1599,415,605,819,975,915,1715,
1223, 1367, 663, 629, 525, 469, 981, 1667, 1587, 1251, 451, 481, 721, 483, 1209, 1457, 415, 1435,
1,1285,823,727,267,833,471,1601,1341,913,1725,2021,1905,375,893,1599,415,605,819,975,915,1715,
1223, 1367, 663, 629, 525, 469, 981, 1667, 1587, 1251, 451, 481, 721, 483, 1209, 1457, 415, 1435,
1,1285,823,727,267,833,471,1601,1341,913,1725,2021,1905,375,893,1599,415,605,819,975,915,1715,
1223, 1367, 663, 629, 525, 469, 981, 1667, 1587, 1251, 451, 481, 721, 483, 1209, 1457, 415, 1435,
1,1285,823,727,267,833,471,1601,1341,913,1725,2021,1905,375,893,1599,415,605,819,975,915,1715,
1223, 1367, 663, 629, 525, 469, 981, 1667, 1587, 1251, 451, 481, 721, 483, 1209, 1457, 415, 1435,
1,1285,823,727,267,833,471,1601,1341,913,1725,2021,1905,375,893,1599,415,605,819,975,915,1715,
1223,1367,663,629,525,469,981,1667,1587,1251,451,481,721,483,1209,1457,415,1435,
1,1285,823,727,267,833,471,1601,1341,913,1725,2021,1905,375,893,1599,415,605,819,975,915,1715,
1223, 1367, 663, 629, 525, 469, 981, 1667, 1587, 1251, 451, 481, 721, 483, 1209, 1457, 415, 1435,
1,1285,823,727,267,833,471,1601,1341,913,1725,2021,1905,375,893,1599,415,605,819,975,915,1715,
1223, 1367, 663, 629, 525, 469, 981, 1667, 1587, 1251, 451, 481, 721, 483, 1209, 1457, 415, 1435,
1,1285,823,727,267,833,471,1601,1341,913,1725,2021,1905,375,893,1599,415,605,819,975,915,1715,
<u>1</u>223,1367,663,629,525,469,981,1667,1587,1251,451,481,721,483,1209,1457,415,1435,
1,1285,823,727,267,833,471,1601,1341,913,1725,2021,1905,375,893,1599,415,605,819,975,915,1715,
迎23,1367,663,629,525,469,981,1667,1587,1251,451,481,721,483,1209,1457,415,1435,
*end 11 */
工, 3855, 4091, 987, 1839, 4033, 2515, 579, 3863, 977, 3463, 2909, 3379, 1349, 3739, 347, 387, 2881, 2821, 1873, 19
$\frac{1}{2}\text{9}, 1929, 2389, 3251, 1149, 243, 3609, 1131, 1701, 143, 1339, 3497, 2499, 1571, 983, 4021, 1625, 3217, 1163, 2977
<u>11</u>, 3855, 4091, 987, 1839, 4033, 2515, 579, 3863, 977, 3463, 2909, 3379, 1349, 3739, 347, 387, 2881, 2821, 1873, 19
isp,1929,2389,3251,1149,243,3609,1131,1701,143,1339,3497,2499,1571,983,4021,1625,3217,1163,2977
 ,3855,4091,987,1839,4033,2515,579,3863,977,3463,2909,3379,1349,3739,347,387,2881,2821,1873,19
59,1929,2389,3251,1149,243,3609,1131,1701,143,1339,3497,2499,1571,983,4021,1625,3217,1163,2977
雌, 3855, 4091, 987, 1839, 4033, 2515, 579, 3863, 977, 3463, 2909, 3379, 1349, 3739, 347, 387, 2881, 2821, 1873, 19
59,1929,2389,3251,1149,243,3609,1131,1701,143,1339,3497,2499,1571,983,4021,1625,3217,1163,2977
壞,3855,4091,987,1839,4033,2515,579,3863,977,3463,2909,3379,1349,3739,347,387,2881,2821,1873,19
59,1929,2389,3251,1149,243,3609,1131,1701,143,1339,3497,2499,1571,983,4021,1625,3217,1163,2977
1,3855,4091,987,1839,4033,2515,579,3863,977,3463,2909,3379,1349,3739,347,387,2881,2821,1873,19
59, 1929, 2389, 3251, 1149, 243, 3609, 1131, 1701, 143, 1339, 3497, 2499, 1571, 983, 4021, 1625, 3217, 1163, 2977
1,3855,4091,987,1839,4033,2515,579,3863,977,3463,2909,3379,1349,3739,347,387,2881,2821,1873,19
59, 1929, 2389, 3251, 1149, 243, 3609, 1131, 1701, 143, 1339, 3497, 2499, 1571, 983, 4021, 1625, 3217, 1163, 2977
1,3855,4091,987,1839,4033,2515,579,3863,977,3463,2909,3379,1349,3739,347,387,2881,2821,1873,19
59,1929,2389,3251,1149,243,3609,1131,1701,143,1339,3497,2499,1571,983,4021,1625,3217,1163,2977
1,3855,4091,987,1839,4033,2515,579,3863,977,3463,2909,3379,1349,3739,347,387,2881,2821,1873,19
59,1929,2389,3251,1149,243,3609,1131,1701,143,1339,3497,2499,1571,983,4021,1625,3217,1163,2977
            /*end 12 */
  if (n < 0) {
    for (j=12; j \le MAXBIT; j++) iv [1+j*MAXDIM] = 1;
                                                  /* Initialize all direction
                                    numbers for the first
                                    coordinate to 1
    for (j=1,k=0;j\leq MAXBIT;j++,k+=MAXDIM) iu [j] = \&iv[k];
    for (k=1; k \le MAXDIM; k++) {
      for (j=1;j<=mdeg[k];j++) iu[j][k] <<= (MAXBIT-j);</pre>
```

```
for (j=mdeg[k]+1;j<=MAXBIT;j++) {
  ipp=ip[k];
  i=iu[j-mdeg[k]][k];
  i ^= (i >> mdeg[k]);
  for (l=mdeg[k]-1;l>=1;l--) {
  if (ipp & 1) i ^= iu[j-l][k];
    ipp >>= 1;
  iu[j][k]=i;
    fac=1.0/(1L << MAXBIT);</pre>
    in=0;
  else
       /* Check if the (n-1)-th number was generated in the previous call
   to sobol. If not, update in and ix */
       if(in!=n-1) {
  unsigned long gray;
\square /* Set ix to 0 */
\bigoplus for (k=1;k<=IMIN(d,MAXDIM);k++) ix[k]=0;
___in=n-1;
gray=in^(in>>1); /* Find gray code of in */
  for (j=1;j<=MAXBIT;j++) {
                            /* Only digits which are 1 are used */
   if(gray&1) {
١,
      im=(j-1)*MAXDIM;
      for (k=1; k \le IMIN(d, MAXDIM); k++) ix [k] = iv[im+k];
    gray>>=1;
TU.
                                 /* Calculate the next vector in the sequence */
       im=in;
      for (j=1;j<=MAXBIT;j++) { /* Find the rightmost zero bit */
if (!(im & 1)) break;
  im >>= 1;
Ī
       if (j > MAXBIT) nrerror("MAXBIT too small in sobseq");
       im=(j-1)*MAXDIM;
       for (k=1; k \le IMIN(d, MAXDIM); k++) {
  ix[k] \stackrel{\wedge}{=} iv[im+k];
  x[k-1]=ix[k]*fac;
       in++;
#undef MAXBIT
#undef MAXDIM
```



```
void halton(int n)
   This is the function halton for generating Halton points.
   It returns the n-th d-dimensional Halton point. The point is implicitly
   returned through the array x. The last two digits of n-1 in base p[j]
   are kept in q1[j] and q2[j]. When both digits become p[j]-1, the
   radical inverse function is computed again. That way the accumulation of round-off error is avoided. In practice, there are not any upper bounds
   on the values of d and n.
   /* actual dimension of the points */
extern int d;
extern int *q1, *q2; /* q1[j] is the last digit of n-1 in base p[j], q2[j] is
              the digit before the last one */
                     /* This will contain the Halton point */
extern double *x;
               /* the first d prime numbers */
extern int *p;
extern int *p_1; /* first d prime numbers minus 1 */
extern double *incr1,*incr2;
                              /* incr1[j] is 1/p[j] and 1/(p[j]*p[j]) */
double find_fi(int p, int n); /* See below */
woid halton(int n)
  double a;
  int j,nn;
static int ins_n; /* The default value of ins_n is 0 */
11 /* Check if the (n-1)-th number was generated in the previous call
     to halton. If not, update q1, q2, and x */

    if (ins_n!=n-1)

T
      ins_n=n-1;
      for (j=0; j<d; j++)
    q1[j]=ins n%p[j];
   q2[j] = (ins n/p[j]) p[j];
    x[j]=find_fi(p[j],ins_n);
      ins_n++;
  for(j=0;j<d;j++)
      if(q1[j]<p_1[j])
    /* It is easy to update when the last digit is less than p[j]-1 */
    q1[j]++;
    x[j]=x[j]+incr1[j];
      else if(q2[j]<p_1[j])
  {
    /* This is the case when the last digit is p[j]-1 and the digit
       before the last one is less than p[j]-1 */
    q1[j]=0;
    q2[j]++;
```

```
x[j] = x[j] + incr1[j] + incr2[j] - 1.0;
      else
    /* This is the case when the last digit is p[j]-1 and the digit
       before the last one is also p[j]-1 */
    q1[j]=0;
    q2[j]=0;
    nn=n/(p[j]*p[j]);
    a=nn%p[j];
    nn=nn/p[j];
    if(nn) x[j] = (a+find_fi(p[j],nn))*incr2[j]*incr1[j];
    else x[j] =a*incr2[j]*incr1[j];
  find_fi(int p, int n)
This returns the radical inverse function fi(p,n) at n for the prime p.
double find_fi(int p, int n)
  int p2, nn;
Udouble s,fi,incr;
 incr=1.0/p;
  p2=p*p;
  nn=n/p2;
fi=0.0;
_s=incr;
  while (nn > 0)
đ
      fi+= (nn%p)*s;
      nn=nn/p;
      s*=incr;
  /* The two largest components of fi are added later to avoid possible
     loss of precision */
  fi=((n/p*p) + fi)/p2;
  fi+=(n%p)/(double) p;
  return fi;
```